Une représentation graphique des schémas XML pour l'enseignement

Emmanuel Desmontils

LINA - Université de Nantes,
2 rue de la Houssinière, BP92208,
44322 Nantes Cedex 03

emmanuel.desmontils@univ-nantes.fr

RÉSUMÉ. *XML est un (méta-)langage actuellement très utilisé. Dans le cadre des formations en informatique, il est indispensable d'initier les étudiants à ce langage et, surtout, à tout son éco-système. Nous avons mis au point un modèle permettant d'accompagner l'enseignement de XML. Il propose de représenter un schéma XML sous la forme d'un graphe mettant en valeur les caractéristiques structurelles des documents valides. Nous présentons dans cet article les différents éléments graphiques et les améliorations qu'il apporte à la modélisation de données en XML.*

ABSTRACT. *Currently, XML is a widely used language. In the context of computer science teaching, it is necessary to introduce students to this language and, especially, at its eco-system. We have developed a model to support the teaching of XML. We propose to represent an XML schema as a graph highlighting the structural characteristics of the valid documents. We present in this paper visual elements and the improvements it brings to data modeling in XML.*

MOTS-CLÉS : *XML, Représentation graphique, Schéma, DTD, XSD, Relax NG, Modèle hiérarchique.*

KEYWORDS: *XML, Schema, DTD, XSD, Relax NG, Graph, Hierarchical model.*

1. Motivation et objectifs

De nos jours, XML (Bray *et al.*, 1997)[1] prend une place importante dans les systèmes informatiques. Ce (méta-)langage est utilisé par exemple aussi bien pour l'échange de données entre Web services, pour le paramétrage d'applications ou pour mémoriser de façon pérenne des informations (par exemple à travers des bases de données XML (Bourret, 1999 ; Gardarin, 2002)).

La complexité des documents XML est extrêmement variable. Même si beaucoup de structures (appelées schémas) sont simples, il est important de bien maîtriser la modélisation de tels documents. Il est souvent utile de s'appuyer sur des méthodologies de conception connues comme UML ou Merise (Carlson, 2001 ; Routledge *et al.*, 2002 ; Gardarin, 2002 ; Desmontils, 2005 ; Lonjon et Thomasson, 2006). Cependant, ces méthodes ne sont pas vraiment satisfaisantes pour les données hiérarchiques.

De plus, pour exploiter ou produire des documents XML, un développeur doit être capable de bien appréhender les schémas. Cela lui permet de tirer profit au mieux de la structure hiérarchique à travers les API dédiées ou les langages adaptés. Nous avons constaté par ailleurs que les utilisateurs de XML n'exploitent pas toujours bien cette structure hiérarchique. Il est donc important d'introduire, dans la formation des développeurs, un enseignement sur XML et son éco-système. Cette formation doit comprendre en particulier :

– les différents langages de schéma (DTD, XSD, Relax NG[2]),

– les principales API de programmation (SAX, DOM[3]),

– les bases de données XML (eXist (Meier, 2003), etc.) et les langages de requête (XPath, XQuery, etc.[4]),

– les langages de transformation (XSLT[5]).

Durant les nombreuses années de formation à XML, nous avons constaté que, pour tous ces outils, le choix d'une représentation graphique permet de mieux appréhender la structure du document à exploiter ou à produire. Nous avons donc recherché une représentation pour mettre en évidence les caractéristiques structurelles du document à valider, en particulier la structure hiérarchique. N'étant pas satisfaits des outils classiques, nous nous sommes inspirés des outils graphiques utilisés pour représenter les modèles relationnels pour proposer notre propre modèle.

Afin d'illustrer notre modélisation, nous avons repris, parmi les nombreux exercices utilisant le modèle, un sujet donné aux étudiants du Master MIAGE de Nantes en octobre 2013. Il concerne la modélisation d'un service (simplifié) de films à la demande auquel est adossé un réseau social spécialisé. Le texte décrivant le contexte, le schéma associé ainsi que le graphe complet qui peut être produit à partir de ce schéma

---

1. Extensible Markup Language (XML) 1.0 : `http://www.w3.org/XML/`
2. XSD : `http://www.w3.org/XML/Schema` ; Relax NG : `http://relaxng.org/`
3. SAX : `http://www.saxproject.org/` ; DOM : `http://www.w3.org/DOM/`
4. `http://www.w3.org/XML/Query/`
5. `http://www.w3.org/Style/XSL/`

sont donnés en section 9[6]. Cet exemple et les illustrations qui suivent utilisent le langage de schéma originel pour XML : DTD. Cependant, l'utilisation de XSD ou de Relax NG ne feraient pas apparaître de différence notable, car nous attachons le plus d'importance à l'aspect structurel des schémas plutôt qu'au typage des informations.

Après avoir présenté les modèles existants (section 2), nous présenterons notre modélisation graphique pour les éléments (section 3), les attributs (section 4) et la structuration (section 5). Après une discussion sur ce modèle (section 6), nous conclurons par le ressenti des étudiants et quelques perspectives.

2. Modèles existants

Les outils de manipulation de schémas dédiés à XML proposent couramment des représentations graphiques pour les schémas XSD ou Relax NG. La figure 1 propose un extrait de représentation graphique typique de schéma XSD[7]. La représentation graphique de Relax NG (quand elle existe) est quasiment identique. Pour les DTD, il n'existe pas de représentation graphique dédiée. Ces représentations ont en commun une structuration sous forme de forêt d'arbres (horizontaux), avec la possibilité de déployer ou non certaines branches. Chaque arbre représente un concept du schéma. Un concept utilisé plusieurs fois apparaitra dans plusieurs branches. Ceci introduit une redondance visuelle importante pouvant amener à des schémas lourds à explorer ("ami" ou "mot-clé" dans notre exemple). Certains symboles sont utilisés sans nécessité, comme le symbole de séquence pour un seul élément (comme "prémium-standard" avec "liste-amis"). On y trouve aussi des symboles peu visibles (le '@' pour les attributs par exemple). La structure en graphe n'est pas clairement visible, le ou les éléments potentiellement racines ne sont pas identifiés et les liens de composition des éléments ne sont pas toujours très lisibles. Relativement faciles à implémenter, ils ne sont pas vraiment utilisables en dessin sur papier.

Les "Feature Models" (Kang *et al.*, 1990) proposent une modélisation hiérarchique spécifique qui a la propriété d'être formalisée (Schobbens *et al.*, 2006). Les symboles sont peu nombreux (4 pour le modèle de base), mais pas vraiment explicites, en particulier pour les cardinalités multiples qui peuvent être explicitées, mais seulement à l'aide de texte (ce qui n'est pas satisfaisant (Moody, 2009)).

Du point de vue général, il existe de nombreux outils graphiques de représentation des modèles conceptuels de données comme le paradigme Entité-Association-Propriété pour la méthode Merise (Tardieu *et al.*, 1983 ; Quang *et al.*, 1991), les diagrammes de classes pour le formalisme UML (Booch *et al.*, 1998), etc. Ces modèles permettent de modéliser n'importe quelle structure de données, mais, de ce fait, ne

6. Les figures de ce rapport ont été conçues à l'aide du logiciel de dessin vectoriel OmniGraffle `http://www.omnigroup.com/omnigraffle`.
7. Oxygen `http://www.oxygenxml.com/xml_editor/xml_schema_editor.html` est présenté ici. Des outils similaires sont présentées sur `http://en.wikipedia.org/wiki/XML_Schema_Editor`.

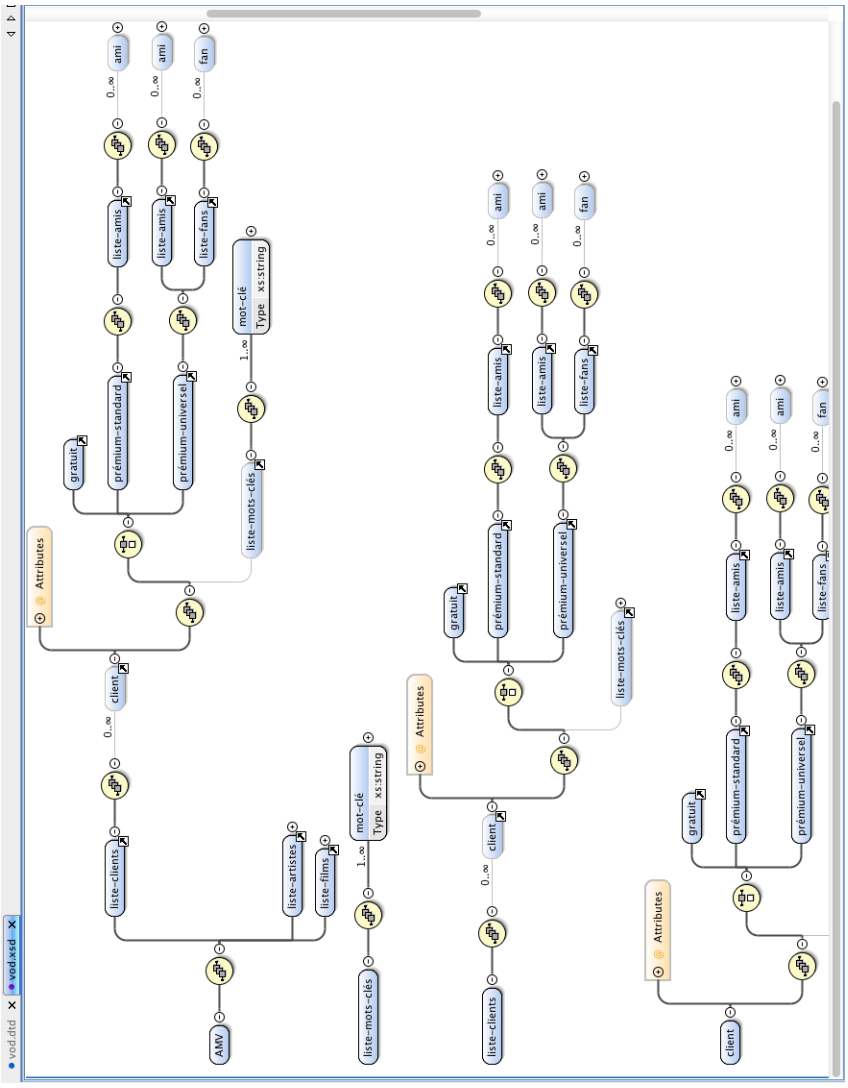


Figure 1. *Visualisation XSD avec Oxygen XML Editor 15.1*

sont pas nécessairement adaptés pour une compréhension des modèles hiérarchiques et ne sont pas faciles à appréhender graphiquement (Moody, 2009).

Parmi ces modèles standard, nous nous sommes intéressés aux "Crow's Foot Diagrams" (CFD) de (Everest, 1976). Ils sont utilisés depuis longtemps en ingénierie de l'information (Martin et Finkelstein, 1988) pour la représentation des tables et de leurs liens dans le modèle Entité-Relation. La forme graphique des cardinalités multiples (1..n ou 0..n) donne son nom à ce type de diagramme. Les CFD sont reconnus

pour leur expressivité visuelle (Moody, 2009) et sont assez familiers, car fréquemment utilisés dans les entreprises. La figure 2 présente un exemple de représentation d'un modèle relationnel avec la notation CFD. Ces pattes mettent en évidence de manière visuelle qu'un étudiant (resp. une matière) sera lié(e) à plusieurs notes.

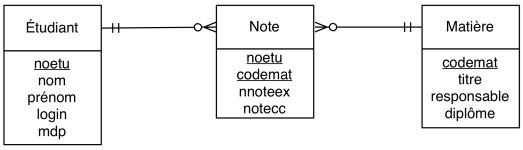


Figure 2. *Exemple de CFD pour le modèle relationnel*

Notre objectif est donc de proposer une représentation graphique pour l'initiation à XML, facile à mémoriser (en limitant le nombre de symboles), facile à comprendre (avec des notations intuitives, comme les CFD) et pouvant éventuellement être utilisée en travaux dirigés (avec papier et crayon) par des novices en informatique. Nous proposons donc un modèle indépendant du schéma XML (DTD, XSD, Relax NG, etc.) utilisant des formes identifiant clairement les concepts manipulés (utilisant des variables visuelles bien distinctes (Bertin, 1983 ; Moody, 2009)), en particulier en adoptant un codage redondant, au sens de (Moody, 2009), sur la forme et la couleur. Ceci permet ainsi d'avoir une représentation visuelle donnant une intuition de la structure hiérarchique des documents XML valides. Ces préoccupations se retrouvent dans (Bihanic *et al.*, 2013 ; Le Pallec et Dupuy-Chessa, 2013) de manière plus générale pour la modélisation du SI dans le contexte de l'ingénierie des modèles. Dans les sections 3, 4 et 5, nous allons détailler les différents composants du modèle.

3. Modélisation des éléments

La notion d'élément XML est centrale, car elle amène le vocabulaire du dialecte et sa grammaire. Dans cette première section, nous nous intéresserons uniquement à des éléments simples. Les éléments forment les sommets du graphe. Les liens entre les éléments correspondent aux contenus des éléments décrits par le schéma. Ces liens seront décrits en section 5.

Un élément est défini en DTD par < !ELEMENT nom-élément contenu>[8] où "contenu" est une expression s'apparentant aux expressions régulières et qui décrit la structure du contenu de l'élément. Dans notre modèle, un élément est représenté par un rectangle vert contenant le nom de l'élément. La figure 3a représente un élément vide décrit par < !ELEMENT like EMPTY>. Certains éléments n'ont pas de contenu structuré : leur contenu est le plus souvent textuel. Dans notre modèle, les textes seront représentés par un rectangle blanc. Éventuellement, ce rectangle contiendra un terme décrivant

8. Ici, comme dans la suite de ce document, les extraits de schéma en DTD seront présentés avec la police Courrier.

le type de texte contenu. La figure 3b représente un élément "like" avec un contenu textuel. Cet élément est décrit en DTD par < !ELEMENT like (#PCDATA)>.

(a) Élément vide (b) Élément à contenu textuel

Figure 3. *Élément*

Les éléments ne sont pas les seules sources d'information en XML : il y a aussi les attributs. Nous allons maintenant nous y intéresser.

4. Modélisation des attributs

XML autorise le positionnement d'attributs associés aux éléments. La liste des attributs est représentée par un rectangle arrondi jaune. Les attributs ont un type (CDATA, ID, etc.) et un comportement (#REQUIRED, #IMPLIED, etc.). La table 1 présente les cas d'attributs les plus fréquemment rencontrés et leur forme dans notre modèle.

	Modèle	**Traduction en DTD**
1	nom-cl	nom-cl CDATA #REQUIRED
2	%date-modif	date-modif CDATA #IMPLIED
3	pseudo	pseudo ID #REQUIRED
4	#client	client IDREF #REQUIRED
5	#(clients)	clients IDREFS #REQUIRED
6	{stars}	stars (0|1|2|3|4|5) #REQUIRED
7	stars/'0'	stars CDATA '0'

Tableau 1. *Formes d'attributs*

Notons que, dans le cas d'une liste de valeurs possibles (ligne 6), notre modèle est incomplet. Ce n'est pas très grave au regard des objectifs pédagogiques de notre modélisation. Cependant, pour être plus complet, il est possible d'écrire "stars$\in${0,1,2,3,4,5}".

La figure 4a représente par exemple un élément vide décrit en DTD par < !ELEMENT client EMPTY>. Il possède trois attributs décrits par :

– <ATTLIST client pseudo ID #REQUIRED>,

– <ATTLIST client nom-cl CDATA #REQUIRED>,

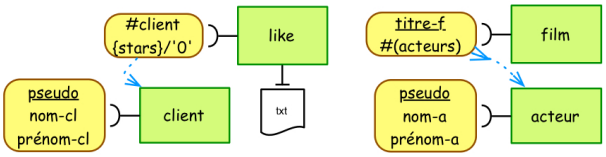
– <ATTLIST client prénom-cl CDATA #REQUIRED>.

Les cas de la table 1 peuvent être combinés. Par exemple, la figure 4b propose un attribut "{stars}/'0'" qui est un attribut pris dans une liste de valeurs (par exemple "0|1|2|3|4|5") et avec comme valeur par défaut '0'.

(a) avec un identifiant "pseudo" (b) avec une référence "client"

Figure 4. *Attributs et identifiants*

Afin de préciser le rôle des attributs de type IDREF(S), il est possible d'ajouter une flèche en pointillé allant de cet attribut vers l'identifiant qu'il est supposé référencer. Les schémas XML ne prévoient pas de préciser ce lien, mais il facilite l'exploitation des documents DTD ou XSD (mise au point des API, utilisation des langages de recherche d'information, etc.). La figure 5a montre le lien entre un attribut IDREF et l'attribut ID correspondant. Ici, l'attribut "client" de l'élément "like" doit contenir le "pseudo" d'un client. La figure 5b représente le lien entre un attribut IDREFS (ici "acteurs") et l'attribut ID qui correspond ("pseudo").



(a) ID ⟵ IDREF (b) ID ⟵ IDREFS

Figure 5. *Lien entre ID et IDREF(S)*

Ces liens ne font pas à proprement parler partie du graphe. Ils sont plus des commentaires, des appuis, pour les utilisateurs du graphe. Maintenant que les noeuds du graphe ont été modélisés, nous allons nous intéresser à la modélisation des arcs.

5. Modélisation des contenus complexes

La structure de graphe apparaît lorsqu'un élément en contient d'autres. Nous allons présenter les différents cas standard et élémentaires que nous pouvons rencontrer.

Tout d'abord, nous allons représenter les opérateurs d'itération "*", "+" et "?". Pour cela, nous nous sommes inspirés des CFD dont l'aspect graphique est plus intuitif que les symboles. La forme de patte sous-entend bien la présence de l'élément en plusieurs exemplaires. Nous exploitons ici (et dans les autres représentations de cette section) les propriétés "physiques" des CFD, en particulier leur perception visuelle (Moody, 2009). La figure 6 présente la représentation utilisée pour chacun des opérateurs. Ces opérateurs permettent de mettre en place les arcs élémentaires entre les différents sommets du graphe.

Il reste deux opérateurs à introduire : la composition d'éléments en séquence et l'alternative. Chacun des éléments qui composent la séquence ou l'alternative font

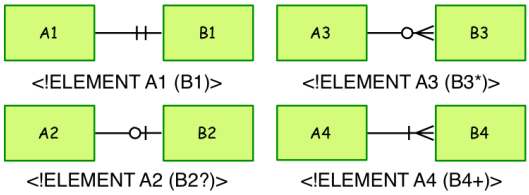
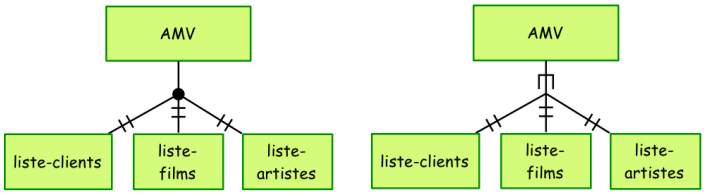


Figure 6. *Itération d'éléments*

l'objet d'un des opérateurs d'itération, d'une autre séquence, d'une autre alternative ou d'un sous-groupe.

La séquence permet de décrire un contenu comme une suite ordonnée d'éléments. Elle est représentée dans notre modèle par un point. L'ordre des nœuds est l'ordre dans le parcours trigonométrique autour de ce point en partant de la gauche du modèle. La figure 7a illustre une séquence et se traduit par :
< !ELEMENT AMV (liste-clients, liste-films, liste-artistes)>.



(a) séquence (b) alternative

Figure 7. *Composition d'éléments*

L'alternative permet de donner le choix entre plusieurs éléments. Elle est représentée dans notre modèle par une fourche. La figure 7b illustre une alternative et se traduit par < !ELEMENT AMV (liste-client | liste-films | liste-artistes)>.

Certaines descriptions de contenu sont construites en utilisant des sous-groupes. Par exemple, supposons qu'un artiste puisse avoir été acteur dans certains films, metteur en scène dans d'autres, voire compositeur de bande originale. Alors, nous pourrions proposer la définition suivante : < !ELEMENT artiste (joue | réalise | compose)+>. La partie (joue | réalise | compose) dans l'exemple ci-dessus est un sous-groupe sur lequel est appliqué l'opérateur d'itération "+". Les trois éléments sont alors répétés dans un ordre quelconque (alternative vue précédemment). Dans notre modèle, un sous-groupe est mis en évidence par un pentagone beige à bords oranges. La figure 8 illustre l'exemple ci-dessus. Cette notion de sous-groupe est "récursive" : un sous-groupe peut lui-même contenir des sous-groupes. Dans notre modèle, il y aura alors une imbrication de zones.

Pour terminer, il est parfois utile d'importer le schéma d'un dialecte spécifique ou d'un langage standard. Dans ce cas, il n'est pas toujours utile de le détailler (impor-

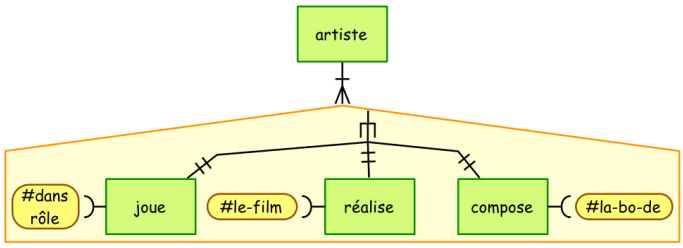


Figure 8. *Sous-groupe d'éléments*

tance secondaire ou, au contraire, parfaitement maîtrisé). Dans ce contexte, il n'est pas nécessaire d'expliciter son graphe. Aussi, nous avons introduit dans notre modèle un symbole représentant un sous-graphe qui n'est pas détaillé. Pour cela, nous utilisons le symbole du nuage. La figure 9a représente la biographie d'un artiste en XHTML[9]. Ce langage, bien connu, n'a pas besoin d'être représenté pour être manipulé. Il suffit alors de donner à "biographie" le même contenu que la balise "<body>" en XHTML (contenu qui est décrit par l'entité paramètre "%body ;").

(a) contenu identique à celui de <body> en XHTML

(b) contenu spécifique, mais non-détaillé pour le moment

Figure 9. *Contenu non-détaillé d'un élément*

Cette simplification peut aussi permettre de ne présenter qu'un graphe partiel. La partie mise en ellipse peut être considérée comme sous-entendue. Le graphe principal est alors considéré comme un graphe hiérarchique, la partie sous-entendue peut faire l'objet d'un graphe ultérieur. La figure 9b représente le contenu de l'élément "liste-films" comme une partie du graphe non développée. Cela permet de réduire la complexité visuelle du graphe et d'introduire la modularisation (Moody, 2009).

L'organisation des éléments dans la page est importante (variables plantaires de (Bertin, 1983)). La disposition des éléments doit, quand c'est possible, rappeler la structure arborescente des documents valides. L'axe vertical permet de représenter la filiation alors que l'axe horizontal représente la fratrie. L'élément supposé racine se trouve en haut du document.

9. La DTD de XHTML est décrite à l'URL : `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd`.

6. Discussion

Ce modèle a été mis au point à des fins pédagogiques. Il n'est donc pas adapté à des schémas très complexes que l'on peut trouver dans certains systèmes d'information. En effet, la topographie du graphe devient difficile à aborder sur une seule page A4 avec un grand nombre d'éléments ou lorsque le nombre des imbrications de groupes dépasse deux ou trois niveaux. La complexité du diagramme devient un frein à sa compréhension. Notre exemple complet (section 9) illustre bien cette difficulté. En effet, il ne comporte qu'une vingtaine d'éléments et semble, visuellement, déjà bien complexe. Par contre, il reste utilisable en conditions opérationnelles pour de petits schémas, souvent présents dans des applications Web à base de services par exemple.

Nous pouvons noter aussi que le graphe seul reste incomplet pour une vue exhaustive des éléments et des attributs. La description du contenu des attributs et des éléments reste très basique. Le modèle ne couvre même pas l'ensemble des types en DTD : les listes ne sont pas détaillées, les types NMTOKEN et NMTOKENS ne sont pas identifiés, tout comme le type NOTATION. Les entités générales ne sont pas non plus explicitées. Par contre, les entités paramètres le sont de manière indirecte (par leurs effets). Dans notre exemple (section 9), le mécanisme d'inclusion de la DTD de XHTML s'effectue par ce principe.

Malgré tout, ces schémas restent un appui de taille pour aborder la construction de chemins de localisation en XPath, pour comprendre le mécanisme de parcours par l'API SAX, les optimisations de parcours avec DOM ou l'application des règles en XSLT. En XPath, par exemple, il est plus facile de construire le chemin pour atteindre une information en se référençant à notre modèle. En particulier, les liens entre les attributs de type IDREF ou IDREFS (une originalité de notre modèle) et leur correspondant de type ID permettent de mieux comprendre les fonctions XPath "id()" (en suivant la flèche) et "idref()" (en remontant la flèche).

De plus, l'utilisation des symboles tirés des CFD permet de donner une idée intuitive de la structure de l'arbre XML résultant. Pour renforcer l'intuition, il est conseillé de travailler aussi sur la topographie du graphe. En effet, dans la plupart des cas, le graphe est déjà un arbre, voir un treillis (comme dans notre exemple en 9), et il est utile de présenter le graphe comme tel. Notons que le cas d'un treillis est intéressant, car il permet de visualiser simplement les éléments pouvant se trouver dans différents contextes (cas de "liste-mots-clés" et "liste-amis" dans notre exemple) en limitant la redondance d'objets que l'on constate dans le modèle classique.

Notre modélisation en graphe peut s'inscrire dans le processus de modélisation de données ou de connaissances sous forme hiérarchique, comme l'illustre la figure 10.

En effet, un processus de modélisation standard commence par l'analyse du problème. Cette analyse amène la construction d'un modèle conceptuel des données, en Merise ou en UML par exemple, puis à un modèle physique, dans notre contexte en XML (Carlson, 2001 ; Routledge *et al.*, 2002 ; Gardarin, 2002 ; Desmontils, 2005 ; Lonjon et Thomasson, 2006) (étapes 1, 2 et 4 dans la figure 10). Notre modèle peut

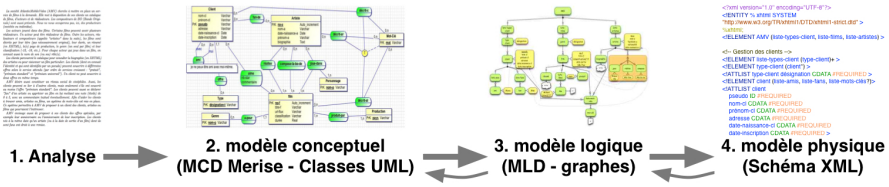


Figure 10. *Processus de modélisation avec notre modèle*

s'insérer entre le modèle conceptuel et le schéma (étape 3 dans la figure 10). En effet, il peut être considéré comme le modèle logique des données, car il va mettre en valeur les propriétés du modèle physique XML (structure hiérarchique et contrôle syntaxique) et réaliser certains types de contraintes énoncées dans le modèle conceptuel. Ainsi, comme dans notre exemple, il est possible de profiter de listes de valeurs courtes (les types d'abonnement par exemple) pour les transformer en contraintes structurelles. De même, les contraintes du modèle conceptuel peuvent parfois être transformées en contraintes syntaxiques dans le modèle logique (Desmontils, 2005). Dans notre exemple, la liste d'amis n'est possible que dans les abonnements "prémium".

Dans le cadre pédagogique uniquement, le processus de modélisation est alors pris à rebours en partant du schéma, objet de l'étude, pour remonter vers la modélisation en graphe pour mettre en évidence la structure.

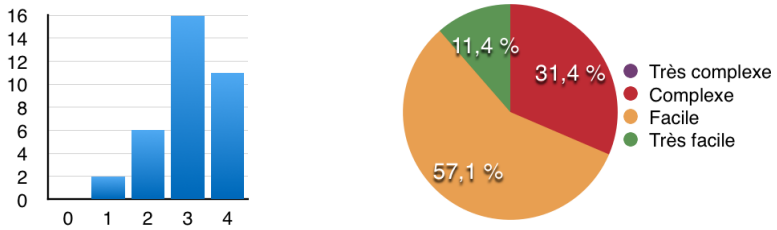
7. Conclusion et perspectives

Le modèle graphique de schémas XML que nous présentons ici est donc un modèle orienté vers l'aspect structurel du document XML et moins vers l'aspect type de données. C'est un modèle intéressant pour l'enseignement, surtout si le schéma n'est pas trop complexe. Il permet une bonne maîtrise du schéma, surtout pour la découverte de XPath, XSLT et des API de programmation (SAX et DOM). Il peut aussi être utilisé lors du processus de construction du schéma XML en jouant le rôle de modèle logique des données (entre le modèle conceptuel en Merise ou UML et le modèle physique qu'est XML).

Le modèle est opérationnel dans le cadre des enseignements de l'éco-système XML en Master MIAGE à l'université de Nantes et en Master CCI depuis quelques années. Cette année, nous avons demandé aux étudiants ayant suivi notre enseignement sur XML (deux modules respectivement au premier et au second semestre) d'évaluer l'apport de ce type de graphe (figure 11)[10]. Notons que ces étudiants n'ont pas eu de présentation détaillée du modèle et que les graphes proposés n'avaient pas de légende. Les concepts ont été présentés au fur et à mesure. Pour les différents exemples et exer-

10. Le détail des réponses à ce sondage peut être consulté à l'URL suivante :
`http://www.desmontils.net/Documents/G4LX/G4LX-2014-03-28.xlsx`

cices donnés en CM, TD, TP et contrôles continus, nous donnions systématiquement le graphe, le schéma (souvent la DTD) et un exemple de document XML valide.



Utilité du modèle (0 = "N'apporte rien" ; 4 = "Très utile")

Maitrise du modèle

Figure 11. *Sondage auprès des étudiants de MIAGE et de CCI de Nantes*

Sur les 52 étudiants, 35 ont répondu (soit plus de 67%). Nous leur avons demandé l'utilité de ce modèle (figure 11a). Ils devaient mettre une note entre 0 (n'apporte rien) et 4 (très utile). 77% ont trouvé le modèle utile (notes 3 et 4) et ils ont donné une note moyenne de 3,02. Nous leur avons aussi demandé la complexité d'apprentissage de ces graphes (figure 11b). Pour 68,6% des étudiants, ils ont été faciles à appréhender. Certains ont trouvé ce type de graphe complexe, en partie à cause du manque d'explications initiales (indiqué dans les commentaires). Les objectifs pédagogiques sont donc en grande partie atteints. Plusieurs pistes d'améliorations sont encore à explorer.

Tout d'abord, nous l'avons déjà évoqué, notre modèle graphique ne permet pas de représenter tous les concepts présents dans un schéma, en particulier pour XSD. Il serait intéressant de mieux préciser graphiquement les caractéristiques (types) des attributs et des contenus textuels sans pour autant augmenter de manière exagérée la complexité visuelle. En particulier, il manque un modèle graphique pour représenter les types complexes et les opérations d'extension et de restriction. Il ne faut cependant pas oublier que l'objectif principal n'est pas de proposer un modèle graphique exhaustif, mais de s'attacher aux propriétés structurelles du schéma à des fins pédagogiques.

Ensuite, à plusieurs reprises, nous avons évoqué la difficulté à présenter un modèle graphique complet et exploitable sans dégrader exagérément la complexité visuelle. Il sera donc intéressant de proposer un cadre général d'analyse permettant de comparer les modèles et de mesurer leurs limites (comme le nombre d'éléments au delà duquel la complexité visuelle sera trop dégradée) en s'appuyant sur les travaux de (Moody, 2009 ; Le Pallec et Dupuy-Chessa, 2013). Ceci permettra d'apporter une aide pour la modularisation des représentations graphiques de schémas XML, mais aussi d'évaluer les schémas eux-mêmes, par exemple en proposant une estimation de l'intérêt pédagogique d'un exercice au regard de la complexité graphique.

Enfin, nous avons montré que notre modèle graphique peut s'insérer dans un processus global de modélisation. Il convient maintenant de mettre en place des processus (semi-)automatiques permettant de passer du modèle physique (schéma) à notre modèle, et réciproquement. De même, il faudra étudier des techniques (semi-)automatiques permettant de passer du modèle conceptuel (MCD Merise ou classes UML) à notre

modèle. Pour cela, nous pourrons évidemment exploiter les travaux effectués en ingénierie des modèles (IDM) (Sendall et Kozaczynski, 2003).

Remerciements

Nous tenons à remercier les étudiants de la MIAGE et du Master CCI de Nantes d'avoir servi de cobaye pour ce travail depuis plusieurs années. Nous voulons aussi remercier P. André, C. Attiogbé et A. Mostéfaoui pour leur soutien et à J.-M. Mottu pour ses relectures et ses conseils. Ce travail a été partiellement financé par le projet ONECAD (Fondation de France, Université de Nantes -Géolittomer UMR 6554 & LINA UMR C6241-).

8. Bibliographie

Bertin J., « Semiology of graphics : Diagrams, networks, maps (WJ Berg, Trans.) », *Madison, WI : The University of Wisconsin Press, Ltd*, , 1983.

Bihanic D., Chevalier M., Dupuy-Chessa S., Morineau T., Polacsek T., Le Pallec X., « Modélisation graphique des SI : Du traitement visuel de modèles complexes », *Actes de la Conférence Inforsid 2013*, mai 2013.

Booch G., Rumbaugh J., Jacobson I., « The Unified Modeling Language (UML) », *World Wide Web : `http://www.rational.com/uml/` (UML Resource Center)*, vol. 94, 1998.

Bourret R., « XML and Databases », 1999.

Bray T., Paoli J., Sperberg-McQueen C. M., Maler E., Yergeau F., « Extensible Markup Language (XML) », *World Wide Web Journal*, vol. 2, n° 4, 1997, p. 27–66.

Carlson D. A., *Modeling XML applications with UML : practical e-business applications*, Addison-Wesley Reading, 2001.

Desmontils E., « Modélisation avec XML, Cours e-miage », 2005, in French, `http://miage.univ-nantes.fr/miage/D2X1/chapitre_modelisationXML/chapitre.htm`.

Everest G. C., « Basic data structure models explained with a common example », *Proc. Fifth Texas Conference on Computing Systems*, Austin, TX, october 1976, IEEE Computer Society publications office, p. 18–19.

Gardarin G., *XML des bases de données aux services Web*, Dunod, 2002, in French.

Kang K. C., Cohen S. G., Hess J. A., Novak W. E., Peterson A. S., « Feature-oriented domain analysis (FODA) feasibility study », rapport, 1990, DTIC Document.

Le Pallec X., Dupuy-Chessa S., « Support for quality metrics in metamodelling », *Proceedings of the Second Workshop on Graphical Modeling Language Development*, ACM, 2013, p. 23–31.

Lonjon A., Thomasson J.-J., *Modélisation XML*, Editions Eyrolles, 2006, in French.

Martin J., Finkelstein C., *Information Engineering*, Savant Institute, Jan 1988, http ://miha.ef.uni-lj.si/predmeti/informatika/Information_engineering.pdf.

Meier W., « eXist : An open source native XML database », *Web, Web-Services, and Database Systems*, p. 169–183, Springer, 2003, http ://exist-db.org.

Moody D. L., « The "physics" of notations : toward a scientific basis for constructing visual notations in software engineering », *Software Engineering, IEEE Transactions on*, vol. 35, n° 6, 2009, p. 756–779, IEEE.

Quang P. T., Chartier-Kastler C., Davison D., *MERISE in Practice*, Macmillan, 1991.

Routledge N., Bird L., Goodchild A., « UML and XML schema », *Australian Computer Science Communications*, vol. 24, n° 2, 2002, p. 157–166.

Schobbens P., Heymans P., Trigaux J.-C., « Feature diagrams : A survey and a formal semantics », *Requirements Engineering, 14th IEEE int. conf.*, IEEE, 2006, p. 139–148.

Sendall S., Kozaczynski W., « Model transformation : The heart and soul of model-driven software development », *IEEE software*, vol. 20, n° 5, 2003, p. 42–45.

Tardieu H., Rochfeld A., Colletti R., Lesourne J., *La méthode MERISE : principes et outils*, Editions d'organisation, 1983, in French.

9. Description de l'exemple

En octobre 2013, les étudiants de Master MIAGE 1ère année ont eu à modéliser en XML les données sur le projet suivant :

La société AtlanticMobileVideo (AMV) cherche à mettre en place un service de films à la demande. Elle met à disposition de ses clients un catalogue de films, d'acteurs et de réalisateurs. Les compositeurs de BO (Bande Originale) sont aussi présents. Nous ne nous occuperons pas, ici, des producteurs (sociétés ou individus). Les acteurs jouent dans des films. Certains films peuvent avoir plusieurs réalisateurs. Un acteur peut être réalisateur de films. Outre les acteurs, réalisateurs et compositeurs (appelés "artistes" dans la suite), les films sont décrits par leur titre (pas nécessairement original), leur durée, un résumé (en XHTML), le(s) pays de production, le genre (un seul par film) et leur classification (-12, -16, etc.). Pour chaque acteur qui joue dans un film, on connait aussi le nom de son (ou ses) rôle(s). Les clients parcourent le catalogue pour consulter la biographie (en XHTML) des artistes ou pour visionner un film particulier. Les clients (dont on connait l'identité et qui sont identifiés par un pseudo) peuvent souscrire à différentes offres selon le service attendu (par ordre de services croissant : "gratuit", "prémium standard" et "prémium universel"). Un client ne peut souscrire à deux offres en même temps. AMV désire aussi constituer un réseau social de cinéphiles. Aussi, les clients peuvent :

– se lier à d'autres clients avec l'offre "prémium standard" ou "prémium universel" ;

– se déclarer "fan" d'un artiste ou apprécier un film en lui mettant une note (étoile) de 0 à 5, avec un commentaire textuel éventuellement, seulement avec l'offre "prémium universel".

Afin d'aider les clients à trouver amis, artistes ou films, un système de mots-clés est mis en place. Ce système permettra à AMV de proposer à un client des clients, artistes ou films qui pourraient l'intéresser. AMV envisage aussi de proposer à ses clients des offres spéciales, par exemple leur anniversaire ou l'anniversaire de leur inscription. Les clients nés à la même date qu'un artiste (ou à la date de sortie d'un film) dont ils sont fans ont droit à une remise.

On peut obtenir alors le schéma (DTD) suivant :

```
<!ENTITY % xhtml SYSTEM
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" > %xhtml;
<!ELEMENT AMV (liste-types-client,liste-films,liste-artistes)>
```

```
<!-- -- Gestion des clients -- -->
<!ELEMENT liste-types-client (client)*>
<!ELEMENT client ((gratuit|prémium-standard|prémium-universel),
                                liste-mots-clés?)>
  <!ATTLIST client pseudo ID #REQUIRED
    nom-cl CDATA #REQUIRED prénom-cl CDATA #REQUIRED
    adresse CDATA #REQUIRED
    date-naissance-cl CDATA #REQUIRED
    date-inscription CDATA #REQUIRED>
<!ELEMENT gratuit EMPTY >
<!ELEMENT prémium-standard (liste-amis)>
<!ELEMENT prémium-universel (liste-amis, liste-fans)>
<!ELEMENT liste-amis (ami*)>
<!ELEMENT ami EMPTY> <!ATTLIST ami avec IDREF #REQUIRED>
<!ELEMENT liste-fans (fan*)>
<!ELEMENT fan EMPTY> <!ATTLIST fan de IDREF #REQUIRED>

<!-- -- Gestion des films -- -->
<!ELEMENT liste-films (genre)+> <!ELEMENT genre (film+)>
  <!ATTLIST genre désignation CDATA #REQUIRED>
<!ELEMENT film (résumé,liste-likes,produit+,liste-mots-clés?)>
  <!ATTLIST film no-f ID #REQUIRED
    titre-f CDATA #REQUIRED classification (0|10|12|16|18) '0'
    date-sortie CDATA #REQUIRED durée CDATA #REQUIRED
    URL-flux CDATA #REQUIRED>
<!ELEMENT résumé (%block;)> <!--%block; contenu de <boby>-->
<!ELEMENT liste-likes (like*)> <!ELEMENT like (#PCDATA)>
  <!ATTLIST like client IDREF #REQUIRED
    date-like CDATA #REQUIRED stars (0|1|2|3|4|5) '0'>
<!ELEMENT produit EMPTY> <!ATTLIST produit pays CDATA #REQUIRED>

<!-- -- Gestion des artistes -- -->
<!ELEMENT liste-artistes (artiste)+>
<!ELEMENT artiste
   (biographie, liste-mots-clés?, (joue|réalise|compose)+)>
  <!ATTLIST artiste no-a ID #REQUIRED
    nom-a CDATA #REQUIRED prénom-a CDATA #IMPLIED
    nationalité CDATA #REQUIRED date-naissance-a CDATA #REQUIRED>
<!ELEMENT biographie (%block;)>
<!ELEMENT joue EMPTY>
  <!ATTLIST joue dans IDREF #REQUIRED rôle CDATA #REQUIRED>
<!ELEMENT réalise EMPTY> <!ATTLIST réalise le-film IDREF #REQUIRED>
<!ELEMENT compose EMPTY> <!ATTLIST compose la-bo-de IDREF #REQUIRED>

<!-- -- Eléments communs -- -->
<!ELEMENT liste-mots-clés (mot-clé+)> <!ELEMENT mot-clé (#PCDATA)>
```

La figure 12 présente un exemple de graphe associé à notre exemple.

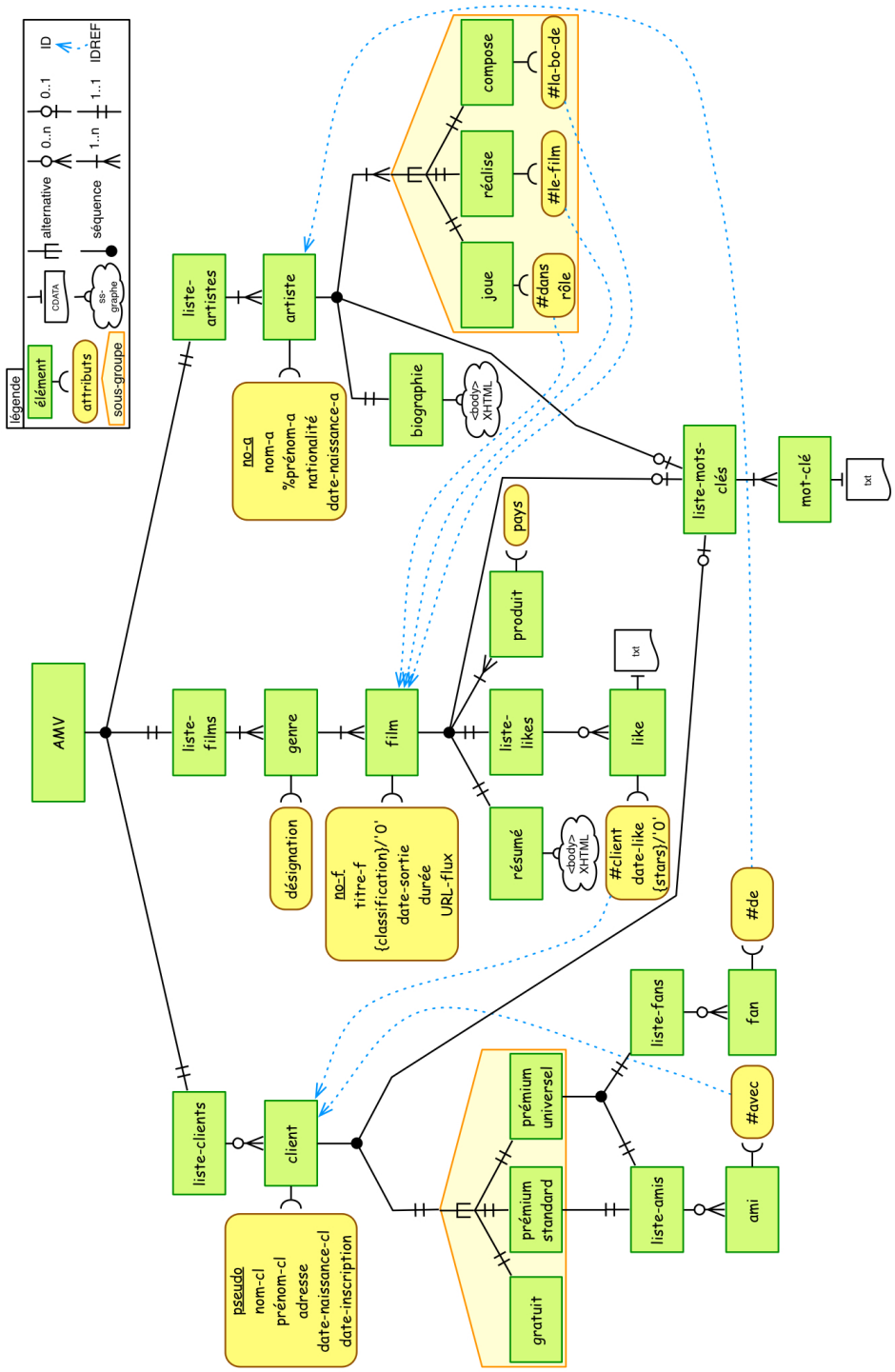


Figure 12. *AMV*